CATS NEWSLETTER

CAPITOL AREA TIMEX/SINCLAIR USERS GROUP

August 1986
Vol. 4, No 5

*** * * CONTENTS * * ***

From the Founder.........1
From the President.......2
Cryptogram...............2
Hi-Res to 80 Columns.....3
Navigating D_File........4
Update on FIND...........5
Why A Fair?..............6
Using the Brother EP-44..6
Translating BASIC #1.....8
Translating BASIC #2.....9
Video comes to CATS.....11
Cryptogram Solution.....11

July Meeting

Newly elected President TOM BENT smoothly and competently conducted the first CATS meeting of his new administraton in the delightfully cool chambers of the New Carrollton library the second Saturday afternoon in July.

The topics covered ranged up and down and all around the Sinclair spectrum (sic!). Tom mentioned the Armstrad takeover and the status of the New QL (the rights to which Armstrad does not own). The availability of a good disk drive will be critical to the QL's success, he said. New cartridges for the QL are available, but somewhat scarce.

A modified 2068, termed a 2048, may end up being produced in Mexico, it appears. FCC approval would be needed for marketing in the conterminous U.S., and will probably never happen.

The saga of the Portuguese SCLD chips seems to be winding down to a happy ending (cf., CATS, May and June, 1986). Special sockets for the SCLD chips will probably be available soon from ELECTRONICS PLUS, one of our faithful advertisers.

LARRY NORRIS of Arlington, VA, joined us for the July meeting. He has been following CATS for some time, now. His interests are photography, magic, and electronics. We were delighted to have him with us again.

MIKE WARMICK explained some of the things he is doing while editing the videotape with which he captured the June (science fair) meeting. The finished product will undoubtedly hold great interest for a number of CATS members.

§§§§

TIM ACORD of Alexandria, VA, shared with us a recent unpleasant happening in his life in the hopes we might gain insight from his experience.

Last month Tim was gone for four days while attending a distant reunion. While he was away, a power outtage occurred. His neighbor used the key he had to enter Tim's house and "clear" the Radio Shack alarm system. In so doing, he inadvertently turned the system completely off.

Sometime later, burglers entered the house and proceeded to carry out all kinds of electronic equipment, VCR's, color TV's, and the like. They even got one of his two T/S 2068's!

Contd →

contd from P1

They might have gotten more, but seemed to be frightened off by something right in the middle of their operations.

Tim is in the midst of dealing with the insurance company right now. Although he didn't keep as good a separate list of his equipment as he would have liked, he did say the insurance people were impressed with the amount of model numbers, names, and serial numbers that he did have. Many of these came from the original boxes he retained for his equipment.

One of Tim's conclusions is that while we can never predict when something like this will strike, the best protection is a good list of the equipment and its numbers, kept separately, but readily available. He admits it sounds like a good job for PRO-FILE!! CHD

Cryptogram

TBVDI CATS PAJHTOUS VB ONU HUSXUPO VDMAAS BHASO XAS ETITBO - VX CAT NEQU EVS PADMVOVADVDI!

Deadlines

Newsletter	Meeting Date
	August 9
August 29	September 13
September 26	October 11
October 24	November 8

Contributors

Tom Bent
Ken Brown
Wes Brzozowski, SINCUS
Hank Dickson
Mark Fisher
Walter Hattery
Bob Howard

Officers & Functionaries

President	Tom Bent
Vice-President	Hank Dickson
Vice-President	Harry Harrison
Treasurer	Ruth Fegley
N/L Editor	Mark Fisher
N/L Production	Sarah Fisher
	Bob Curnutt
Corresp Secy.	Mike Cohen

Presidential Ramblings

Here we are at mid-summer and all is calm. The last meeting was the usual small summer group and involved a lot of rambling about the future of Sinclair computing (we don't see any problems though). We discussed the latest words on the Amstrad buy-out, the introduction of the Thor (QL clone) and possible other entries by interested QL cloners.

We has the 2068 wired up to a Sinclair RGB monitor (courtesy of Ruth Fegley) and everyone naturally awed at the clarity and legibility of the 2068. RGB monitors are indeed great! We also showed the QL doing its "thing" by multitasking several programs very handily. Unfortunately, I forgot to bring the disk cable and couldn't demo what I had intended.

SPECTRUM IF/1 DEMO

The next meeting holds a lot for everyone. Dave Rothman has promised to bring in his Spectrum and Interface 1 and demo some microdrive based software and talk about converting cassette based software to disk (or rather microdrive). Let there be no doubt that the software he will demo will indeed be eye opening!! I have already previewed this stuff.

The hardware sessions have finally simmered down. I suppose everyone who originally started with the sessions has upgraded to the point of contentment. I have been impressed recently with the type of computer questions that have been put to me. It goes without saying that the level of computer literacy and knowhow in the club as a hole has increased considerably.

Congratulations to all of you!!

We are now looking for input from you as to what or how you would like to see the morning session to develop.

If you can't make it to the meeting, then drop CATS a postcard with your ideas. Tom Bent

Sending Hi-Res to an 80 Column Printer

By Mark Fisher and Bob Howard

The modern dot-matrix printer is a marvel of flexibility. It can be treated as a super-flexible typewriter - changing type fonts on command and summoning up an almost endless array of pre-defined graphics characters. But there are some times when you want more: either a specific character that the printer doesn't "stock," or perhaps the ability to put a full hi-res image on the paper.

Almost every dot-matrix printer is capable of producing hi-res images. This is done in two steps:

1) Tell the printer how many bytes of hi-res information are coming. This is necessary because there is no intrinsic difference between a byte intended to define an ASCII character and a byte intended to define one column of graphics information. The specific commands vary for each printer - for the Itoh 8510 it's ESC S (four digit ASCII numeral). For the SG-10 it's ESC K (two bytes, in low-high format).
2) Send the information to the printer. Each byte sent will only control one column of pixels on the page. Each individual pin of the print head will be controlled by a particular bit position in the data byte. The chart sent in by Bob Howard summarises the possible variations between printers.

Now I'll try it. Since I'm using an Itoh 8510, I'll send ESC S 0008, followed by eight data bytes; 28, 34, 85, 81, 85, 34, 28, 0. ⊖

THA-THA-THA-THAT'S ALL, FOLKS!

⊖ ⊖ ⊖ ⊖ ⊖MF

PS. Kicking your printer into graphics mode will let you inspect the binary output of your interface directly - useful for debugging interface-printer setups.

TABLE 2-1
Three Families of Dot-Matrix Printers

PIN
A02
→

IBM-EPSON	APPLE	TANDY
● 128 ● 64 ● 32 ● 16 ● 8 ● 4 ● 2 ● 1	● 1 ● 2 ● 4 ● 8 ● 16 ● 32 ● 64 ● 128	● 1 ● 2 ● 4 ● 8 ● 16 ● 32 ● 64
IBM Graphics Printer ~ Epson FX-80, FX-100 Epson RX-80, RX-100 Epson MX-80, MX-100 ~ Star Gemini 10, 15 ~ Mannesmann Tally MT160 Mannesmann Tally Spirit ~ Inforunner Riteman ~ Panasonic KX-P1090	Apple Imagewriter Apple Dot Matrix ~ C. Itoh Prowriter I C. Itoh Prowriter II ~ NEC 8023A ~ Anadex DP-9000A, 9500A Transtar T315 ~ Canon A-1200 ~ Personal Micro DMP-85	TRS-80 LPVIII TRS-80 DMP-110, -120 TRS-80 DMP-200 TRS-80 DMP-400, -420, -500 TRS-80 CGP-220 ~ Okidata Microline 82, 93 ~ IDS Prism 80, 132

Navigating Through the Display File Jungle

By Wes Brzozowski
SINCUS News, May/June 1986

At least one reader was disappointed with my April Fool article on display files, last issue. He'd first thought that someone finally got around to making the whacky display file more navigable. Well, we can do that too!

Those who've manipulated the nice orderly display file on the 1000 have been somewhat perplexed by the TS 2068. Yes, the illustration on page 251 of the user's manual does explain it. Still, most everyone reacts with the question, "Why ?"

This configuration was designed to dovetail with the Z-80 instruction set to allow us some reasonably fast display file operations. But only if we understand how.

If you've ever watched the distinctive way that a SCREEN$ is LOADed into the computer, 8 character lines at a time, you've actually seen the odd display layout. If you've ever taken an elementary digital design course, you may also have noted the similarity between that pattern and the nice regular spacing of some variables on a Kavnaugh map.

This might lead you to suspect that the strange display file layout is merely due to the rearranging of several address lines in the display circuitry. If so, you'd be correct. What this means is, we can start with a nice orderly row-column notation, and transform it into the display file notation simply by swapping several bits. But why did they deliberately miswire the display hardware?

There are some good reasons, honest! If we look at some of the computers that the "big kids" use, we see that they have both text and graphic display modes. These modes are set in the display hardware. The graphics modes are, of course, simply to allow us to do graphics work. The text modes, however, relieve the computer of having to figure out which pixels to set in order to put a particular character on the screen. That's all done in hardware. This buys us speed, at the expense of extra hardware. While the speed increase may not be noticeable when just a few words are put on the screen, there's a definite difference when an entire screen of text is put up.

We might say that the 2068 has no text modes, only graphics modes. The "text modes" are simulated in software - the 2068 has to figure out which pixels need to be set in order to put a character on the screen. This means that text operations on the screen will be somewhat slow.

To compensate, the designers used every trick they had. I'll show you the computations needed to support a "text mode," and the tricks that were used to speed up the process.

Many readers are aware that the "pixel patterns" for the characters are stored in a table at the top of ROM, 8 bytes per character. Because we have the option to produce our own pixel patterns, the system variable CHAR$ "almost" points to this table.

Now, to find the pixel pattern for a particular ASCII character, we do the following things:

1. Put the ASCII value in a register PAIR.
2. Shift the value left 3 bits.
3. Add the value in CHAR$ to this.

And we have the address of the pixel pattern! Here's why: The pixel patterns are arranged in order of their ASCII value. The first 8 bytes are the pattern for a blank space, which is 32d, and so on. The left shift multiplies everything by 8, since each character takes up 8 bytes. Since the first character's value is not zero, but 32, we would expect to have to subtract 256 (8*32) from this result to get the starting address. But the value in CHAR$ is already 256 counts lower than the table, so the subtraction doesn't have to be done! This saves us some time in computing the address. It's not a lot of time, but every little bit helps.

Once we have the address of the 8 bytes of the pixel pattern, we have to figure out where in the display file to put them. Before we do this, please read over the description in the box on going from a pixel row - column notation to the display file notation. This starts us with a pixel row and column, and transforms them into a memory address and a bit position within that byte in memory. Note that my pixel notation is not the same as that used by PLOT and POINT in BASIC; see p. 152 of the user's manual. BASIC has to transform its own notation into the one I show here, and THEN transform that into a display file notation (silly but true).

Finding the display file location of the topmost byte of the character is easily done when you understand the description in the box. However, doing the same thing eight times for each character would waste a lot of time. Fortunately, when you know where one pixel line for a character is located, it's very easy to fine the next seven. Look in the box at the 2068/SPECTRUM display file format. By adding 256 to the address for a given pixel line, we'll get the address of the pixel line right below it, because we've just incremented the "pixel line in character" field. We can only do this until the field becomes 111 binary (or 8 decimal), but at that point we're at the bottom of the character, and don't WANT any more.

Suppose the display file had been laid out in a more orderly fashion? The "pixel line in character" field would be swapped with the "line in 8 group" field. That's all. Then to find the address of the next pixel line we'd just add 32 decimal, instead of 256. But this is NOT as convenient.

If the address were in the HL register, for example, we could add 256 simply by doint an INC H. Adding 32 would require us to ADD some other register to HL. This not only takes longer, but ties up an additional register pair. This is even worse, because it turns out we can't easily use the speedy DJNZ instruction for a test loop if we tie up any other registers. It may not be obvious until you try to write the code yourself, but it's so. As things are, we have just enough registers.

While the ROM display routine uses this trick to gain speed, it loses it again, for other reasons. This is because the same routine also handles the standard graphics characters, user defined graphics tokens, the TS 2040 printer, things like INK, BRIGHT, INVERSE, FLASH, and moving the character position in response to AT and TAB. It wastes a lot of time checking what it's supposed to be doing, and then saves a little time when it finally gets around to doing it.

So what's the point? Programs like word processors tend to contain their own custom display routines that run VERY fast. The information given in this article is suufficient for the enterprising machine code programmer to write his or her own, and if you've been trying to put data directly into the display file, this might just be what you've been looking for. This will be of little help to those who want to PEEK and POKE the display file from BASIC. The BASIC commands for operating on the display file probably will work faster and certainly a lot easier. Also note that even in machine code, this odd display file layout will force graphics routines to run slower than they otherwise might, since you have to swap address bits to get the row-column position you want.

But what it does, it does well. While it sacrifices some graphics speed, it gives us a faster text display. In a way, it "narrows the gap" between them. There will be some users who lose out by this design decision, but I think the majority of us would be much more irritated by slower text. Oh well, you can't please everyone.

Questions about this (or any other TS 2068 stuff)? Write me, Wes Brzozowski, 337 Janice St., Endicott, NY 13760. Please include a stamped, self addressed envelope for a reply. Since I sometimes get swamped you might have to wait a bit, but I will get to you. If you're in a hurry, call me at (607) 785-7007. Try to call before 9:30 PM Eastern time. Hope to hear from you!

Update on FIND

Dear CATS Members,

Recently, I submitted a program to use as the "FIND" utility found in most EDITors--but not in the T/S 2068, to the CATS Newsletter. To those of you that have successfully tried it, I hope you found it to be useful and, to those who could not get it to work, my 'phone number is 439-8341(at work: 394-3597). Perhaps I can help you get it running.

To those of you who would would like to use it with the Spectrum ROM it will work just fine if Line 9987 is changed to:

```
9987 DATA 190,32,180,35,14,0,19,
16,229,193,193,205,27,26,62,32
```

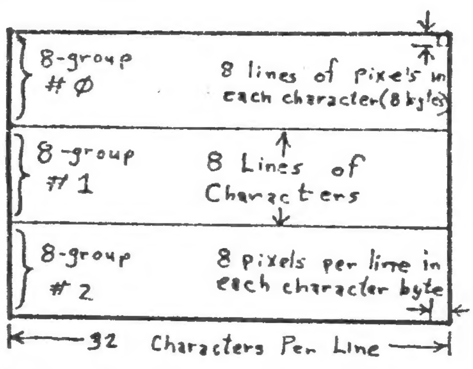
In other words, the "136" and the "23" in Line 9987 should be changed to "27" and "26" respectively. Good luck!

SINCERELY YOURS,

Walter V. Hattery

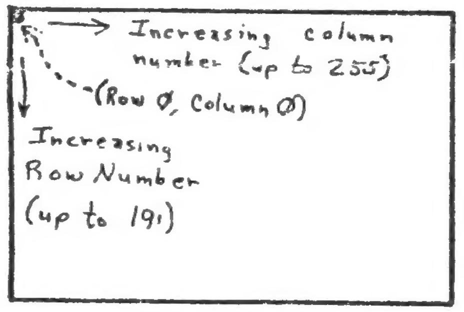
Going From Pixel Row-Column Notation to Display File Notation

Try to picture the screen as:



8-group #Ø
8 lines of pixels in each character (8 bytes)
8-group #1
8 Lines of Characters
8-group #2
8 pixels per line in each character byte
32 Characters Per Line

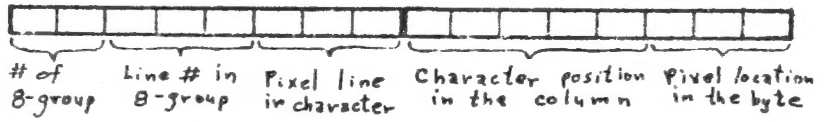
But also try to picture it as:



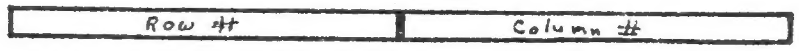
Increasing column number (up to 255)
(Row Ø, Column Ø)
Increasing Row Number (up to 191)

Both of these notations exist simultaneously

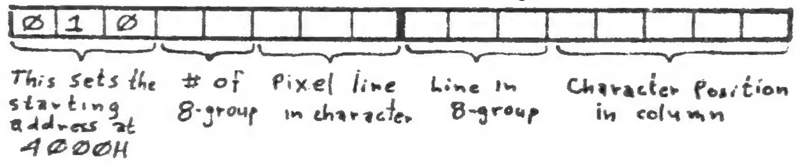
It would be nice to specify a pixel in a double register as:



Because it would be identical to:



But the TS2068/Spectrum Display file is like this:



Ø 1 Ø
This sets the starting address at 4ØØØH
# of 8-group
Pixel line in character
Line in 8-group
Character Position in column

Wes Brzozowski 4/23/1986

So, to transform row-column notation to display notation;

1.) Shift right 3 bits, to eliminate "pixel location in byte"

2.) Put "Ø1Ø" in the upper 3 bits

3.) Swap the "line # in 8-group" and "pixel line in character" fields at their NEW positions

4.) This gives the address of the proper byte. Now use the "pixel location in byte" field to set, reset, or test the proper bit.

Why a Fair?

In June CATS had a "Salute to Science Fairs".

Often people have asked, "What good is a science fair, anyway?"

A convincing answer is hard to make in anything but abstract terms. One specific case does come to mind, however, which might be illuminating.

Charley Dickson, with his sister Cora, founded a predecessor Sinclair users group which eventually evolved into CATS.

In 1980, when he was an eighth-grader in junior high school, he entered his first science fair. It was a "show-&-tell" about electricity, something which greatly fascinated him.

In high school he moved on to combine electronics and physics for the science fair, analyzing certain acrade game hardware. Later he did a project on pin-hole optics in space, based on his work as a student volunteer at NASA's Goddard Space Flight Center.

In between he created some electronics for exhibits at the school system's Howard B. Owens Science Center and designed the command and control circuits for an Explorers' NASA get-away special (GAS) project.

Next December he should become a senior in the School of Electrical Engineering, UM College Park.

The germ of most of this goes back to his 8th grade science fair project, the determination to both learn from--and achieve in--the science fair arena, and the patient encouragement of teachers, friends and family who saw something worthwhile in all the mess and commotion.

It is the intention of CATS--in its own small way--to continue to encourage young people in scientific endeavors, particularly when they show the creative use of low-cost computers in working towards scientific objectives.

It is not at all impossible that someday we may be fortunate enough to recognize the spark of creativity in some student who will, for the good of us all, produce a contribution illuminating our whole society for years to come.

This is Tasword II used with the Tasman Serial Interface and the Brother EP-44 printer.

The word processor naturally uses any of the keys on the 2068 that are labelled. I have been wondering for some time if it was possible to access the special keys on the typewriter through the ASCII codes and ESC or CHR$27.

The standard decimal codes for the numbers 0 to 9 are 48 to 57, 58 to 64 are : ; < = > ? @ which are accessible by just typing in those characters using symbol shift.

By redefining the printing codes in the Tasword system it is possible to access the special characters on the typewriter. It the following I have redefined the graphics on keys 1, 2, 3 and 4 to produce the special characters:

1 = Ø 2 = ß 3 = µ 4 = Ä

It is also necessary to set EP-44 for 8 Bit Code for most of the characters. For example the normal characters decimal 160 to 174 are ! " # $ % & ' ( ) * + , - . / are redefined in the 8 bit mode as follows:

á 160	170	Ç 128	ï 139	û 150	± 241
í 161	½ 171	ü 129	î 140	ù 151	÷ 246
ó 162	¼ 172	é 130	ì 141	§ 152	° 248
ú 163	¡ 173	â 131	Ä 142	Ö 153	² 253
ñ 164	« 174	ä 132	Å 143	Ü 154	
Ñ 165	» 175	à 133	É 144	¢ 155	
ª 166		å 134	æ 145	£ 156	
º 167	ß 225	ç 135	Æ 146	¥ 157	
¿ 168		ê 136	ô 147	₧ 158	
⌐ 169	µ 230	ë 137	ö 148	ƒ 159	
		è 138	ò 149	Ø 237	

By defining those you want as "graphic symbol 27 dec code" for the graphics you can use them in your wordprocessor text to be printed on the EP-44.

BOB HOWARD, WA6DLI

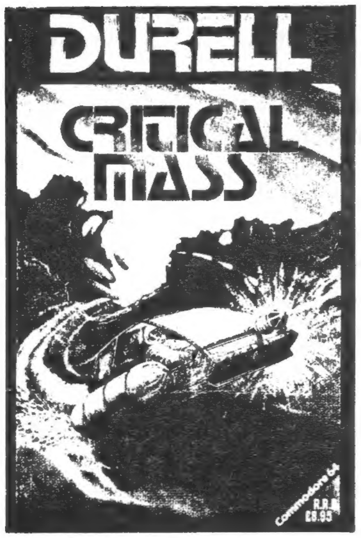
KNIGHTED COMPUTERS
707 HIGHLAND ST
FULTON, NY 13069
(315)593-8219

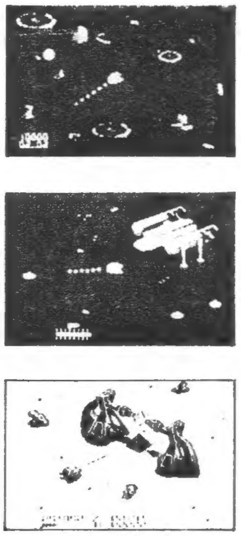
VISA

MasterCard

- A N N O U N C I N G -

NEWLY CONVERTED SOFTWARE FOR YOUR ENTERTAINMENT

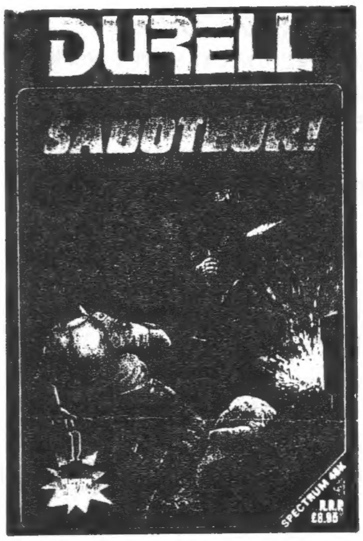


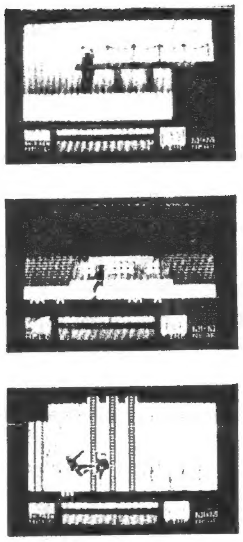


CRITICAL MASS (Arcade)

An outlying system of the Terra Federation has set up an advanced anti-matter conversion plant on a centrally positioned asteroid to supply the local colonists with energy. A surprise attack by alien forces has successfully overcome the asteroid's defences and the aliens are now threatening to self-destruct the power plant unless the colonists offer an unconditional surrender. The self-destruction process would effectively turn the power plant into a massive black hole that would wipe out the entire planetary system along with a number of nearby stars. Unconditional surrender offers an equally horrific prospect. Your mission is to infiltrate the enemy position and disable the anti-matter plant before the aliens achieve CRITICAL MASS.

CATALOG #1026 $15.95





SABOTEUR (Martial Arts Arcade Adventure)

You are a highly skilled mercenary trained in the martial arts. You are employed to infiltrate a central security building which is disguised as a warehouse. You must steal a disk that contains the names of all the rebel leaders before its information is sent to the outlying security stations. You are working against the clock, both in getting to the disk, and in making your escape. You must enter the building from the sea by rubber dinghy, and will then only be able to leave by the helicopter on the roof. You will have to find your way around the warehouse, making use of the guards' own weapon supplies in your struggle through overwhelming odds.

CATALOG #1098 $15.95

- - - - - NEW ACCOUNTING SOFTWARE - - - - -

ACZ GENERAL LEDGER - "A small business accounting system"

If you've been thinking your small business accounting could be done faster, more easily and economically with a T/S2068 - you're right! The ACZ GENERAL LEDGER is a complete ledger and financial reporting system specifically designed to meet the accounting needs of a small business.

The ACZ GENERAL LEDGER is a true double entry system that has the benefits of a manual ledger without the drawbacks. The program identifies transactions that do not balance and gives you the opportunity to make corrections. It will not post an out-of-balance transaction. Up to 150 named accounts, and up to 800 entries and accounts each month. Prints the following reports: Year-to-date Income Statement, Monthly Income Statement, Balance sheet, Trial Balance, Ledger Detail, Journal Entries, and Chart of Accounts. 20 page Instruction manual included.

CATALOG #1139 $19.95

PLEASE INCLUDE $3.00 - SHIPPING & HANDLING VISA/MC ACCEPTED

Thanks for your PAID AD

Translating BASIC Programs

By Ken Brown
Reprinted from The Computerist, Jan '83
(Predecessor to the CATS Newsletter)

Sooner or later, as you browse through computer magazines and books, you will find a published program that you like - only to discover that it won't work in Sinclair BASIC. This problem arises because there is no standard form of the BASIC language. At this point you have two choices: Try to write the program yourself, from scratch, or try to "translate" the program into Sinclair BASIC.

Some differences between BASIC languages are easily remedied. For example, some computers allow the use of multiple assignment statements such as:

```
        10 A=B=C=D=5.4
```

Sinclair BASIC does not allow this type of statement, although it can be easily translated into:

```
        10 LET A=5.4
        11 LET B=A
        12 LET C=A
        13 LET D=A
```

Not as compact, but equally functional. [Ed note: Sinclair BASIC will accept the original line, as long as a LET precedes the numbers. Unfortunately, it will always assign A a value of zero, as it assumes that the multiple equals signs mean the writer wants a logical evaluation of the expression (((B=C)=D)=5.4). As logical values are either 0 or 1, they can NEVER equal 5.4, and thus A will always take the "not true" value of 0.]

Other differences, such as the lack of a READ or DATA statement in Sinclair BASIC, are more difficult. I'll try to tackle READ DATA in this installment, and cover other differences in a later installment.

While READ and DATA are two separate BASIC commands, they are used together. A DATA statement contains a list of data items, either numbers or strings, usually seperated by commas. When the computer encounters a DATA statement it is ignored, much in the same way a REM statement is ignored. However, when a READ statement is encountered, the computer searches for the first DATA statement in the program and assignes the first data item in the DATA statement to the variable in the READ statement.

Here is an example:

```
        10    DIM A(5)
        20    DATA 5,6,-2,0,3
        :
        100   FOR I=1 TO 5
        110   READ A(I)
        120   NEXT I
```

The first time line 110 is executed the computer locates the DATA statement in line 20, locates the first data value (in this case 5), and assigns it to A(1). The computer remembers that the first value has been "used" and the next time line 110 is executed the computer assigns a value of 6 to A(2). Every time a READ statement is encountered the computer will take the next value from the DATA statement. Once the data items in the first DATA statement have been used the next READ will cause the computer to search further on in the program to find the next DATA statement. A RESTORE 20 command is used to move the computer's "pointer" back to the start of line 20.

Now, can you translate the above example into Sinclair BASIC? One approach, the brute force technique, would be:

```
        100   LET A(1)=5
        110   LET A(2)=6
        120   LET A(3)=-2
        :
        140   LET A(5)=3
```

This works for a small array, but suppose that the array is to contain 100 values. The following method is more elegant and tends to preserve the appearance of the original program:

```
        10    DIM A(5)
        20    LET D$=" 5, 6,-2, 0, 3"
        :
        95    LET POINTER=1
        100   FOR I=1 TO 5
        110   LET A(I)= VAL D$(POINTER
      TO POINTER+1)
        115   LET POINTER=POINTER+3
        120   NEXT I
```

Study this example carefully to see how it works. First the DATA statement is replaced by a string variable D$. The commas are not required, though they do make the line easier to read. The blank spaces, on the other hand, are necessary. Because one of our data entries (-2) is two spaces long, making each entry two spaces long makes the "READ" part of our example easier to do. The VAL function takes the portion of D$ from POINTER to POINTER+1 and converts it to a number. [Ed note: for assignment of string variables, the above technique is easier to do than the cumbersome handling of string data in the "real" READ/DATA statements of the TS 2068.] Line 115 then moves the POINTER to the next item on the list, as the READ statement would do automatically. Note that because of the comma the pointer is incremented by three, even though each item is only two spaces long.

In this example the READ statement was used to assign values to an array. This is probably the most common use of the read statement. Occasionally, a program will use the READ statement in a slightly different way. However, it should still be possible to use the ideas I've presented here to translate the program into Sinclair BASIC.

Next chapter: Multiple statement lines and computed GOTOs.

Translating BASIC Programs

By Ken Brown

From the Computerist #6 (Feb. '83)

Multiple Statement Lines

Many computers allow more than one BASIC statement to be included in a single line of a program. Each statement, normally seperated by a colon or a slash (/), is executed in turn starting with the first statement after the line number. In most cases, translating such a program simply requires assigning each statement an individual line number.

Here is an example:

```
130  LET A=2:LET B=3:LET C=A/B
```

When this line is executed A is assigned a value of 2, then B is assigned a value of 3, and finally C is assigned a value of 2/3. Translated to Sinclair BASIC this line would look like:

```
130  LET A=2
131  LET B=3
132  LET C=A/B
```

One difficulty in translating this type of line occurs when the lines in the original program are sequentially numbered. In this case the line numbering in the Sinclair program may be very different than in the original. In this case GOTO and GOSUB statements must be examined carefully.

```
130  LET A=2:LET B=3:LET C=A/B
131  LET A=A+1:LET D=C/A
 :
325  GOTO 131
```

In this program line 325 causes the computer to jump back to line 131 and begin execution with LET A=A+1. When this program is translated the line number of the statement LET A=A+1 will be changed, and the GOTO statement in line 325 must be changed to reflect this.

```
130  LET A=2
131  LET B=3
132  LET C=A/B
133  LET A=A+1
134  LET D=C/A
 :
325  GOTO 133
```

In a short program it is easy to locate all of the GOTO and GOSUB statements and change them if necessary. However, in a program of several hundred lines just keeping track of this type of change can become a monumental task. If the lines in the original program are incremented by 10's, or if only a few lines contain multiple statements then translation is probably possible. If, on the other hand, large portions of the program are sequentially numbered and contain multiple statement lines the program should probably be avoided.

There is one situation in which translating a multiple statement line becomes more difficult. This occurs when the line contains an IF...THEN statement. Unfortunately, computers which allow multiple statement lines handle this situation in two different ways. I believe that the method I will present first is the most common.

Here is an example of this type of line:

```
100  IF A=0 THEN PRINT "HELLO":LET A=1:LET B=-1
110  LET C=A
```

When the condition contained in the IF...THEN statement is true (in this case when A=0) then all of the statements after the IF...THEN statement will be executed. If, on the other hand, the condition is false then none of the statements after the IF...THEN statement will be executed, and the computer jumps immediately to the next line in the program.

In this example, if A=0 when the computer reaches line 100 then the message HELLO will be printed, A will be assigned a value of 1, and B will be assigned a value of -1. If A is not equal to 0 when the computer reaches line 100 the computer jumps immediately to line 110. No message will be printed, and the values of A and B will not be changed.

Translating this example to Sinclair BASIC requires some thought. Here is the same example translated to work properly:

```
100  IF A NOT =0 THEN GOTO 110
101  PRINT "HELLO"
102  LET A=1
103  LET B=-1
110  LET C=A
```

Although the IF...THEN statement in line 100 has been changed considerably these lines function in exactly the same way as the multiple statement lines in the previous example. When the computer reaches line 100 if A=0 then the new condition in the IF...THEN statement is now false. The computer jumps to line 101 and executes the statements which previously followed the IF...THEN statement. If A is not equal to 0 the new condition is now true and the statement GOTO 110 is executed. This skips the statements at lines 101 to 103.

Here is a general procedure for handling multiple line statements containing IF...THEN statements. If there are any statements in the line which proceed the IF...THEN statement these are treated in the normal way. The condition in the IF...THEN statement is changed so than the new condition is the opposite of the original (= becomes either NOT = or <>, etc.). Rewrite the IF...THEN statement so that it branches to the next line from the original program. Finally, place the original statement contained after the THEN clause, as well as all of the other statements to the right of the IF...THEN statement in the original line, on individual lines after the IF...THEN statement.

While most computers which allow multiple statements handle IF...THEN statements in this way, there are apparently exceptions. In these computers when the IF...THEN statement is false the computer jumps to the next statement in the line (the statement after the colon). If this is the case translating a program only requires writing each statement on a seperate line. However, unless the program description explains how this situation is to be handled, it is probably safest to assume that it is handled in the first way.

Translating programs containing multiple statement lines may be difficult at times. There is, however, a wealth of software avalible in books and computer magazines which is written for computers which allow multiple statement lines. In fact, the new Sinclair Spectrum allows multiple statement lines. Find a small program and try it. When you succeed you'll be ready to tackle a large one.

Your One-stop Electronic Parts Store

Special to CATS
5% off any purchase if you bring your n/l (with name attached)

Thanks for your PAID AD!

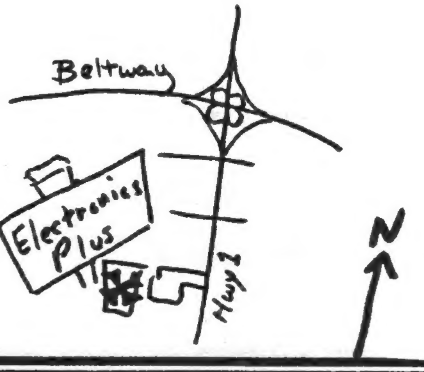
Full line electronic components and test equipment.

Parts and equipment for the radio amateur and enthusiast.

Reasonable prices.

Credit terms available for recognized companies.

Hours: Mon - Fri: 9AM - 6PM
Sat: 10AM - 3PM
Sun: Closed



ELECTRONICS PLUS

9600 Baltimore Blvd., College Park, Maryland 20740 (301) 441-9009

A CATS First

Last June 14th, for the first time in its five-year history, CATS had its entire meeting caputured on videotape!!

Engineering this Guiness-class event was CATS member MIKE WARMICK who was ably assisted by WILLIE USHER, another CATS member, who is a student at Eastern high school.

Mike has become active in a group called "Acting for Television". It is a not-for-profit TV acting organization designed to help groups, classes and individuals become acquainted with the demands and disciplines of the television medium.

Besides having access to the equipment of the group, Mike has been acquiring some impressive studio-quality hardware of his own. He has been filling up his schedule creating video's covering business and family events, sports spectaculars and cultural happenings.

The CATS meeting taped by Mike and Willie could be easily broadcast by the public access channel of any local cable company, its quality is that high. The tape begins with the electronics group meeting and includes the science fair presentations and Tom Bent's demonstration of the newly-augmented Quantum Leap (QL) computer.

If you would like information about how you can have your event captured on tape, give Mike a call at: (202) 398-3761.

In the meantime, Capital PC Users Group:

EAT YOUR HEART OUT!!!

Crypotgram Solution
(honest)

!GNINOITIDNOC RIA EVAH UOY FI -
TSUGUA ROF TROPS ROODNI TCEFREP
EHT SI RETUPMOC RUOY GNISU

Capitol Area Timex/Sinclair Users' Group
P.O.Box 725
Bladensburg, MD 20710

Name ____________________
Address ____________________
____________________ ZIP ________
Phone Home ____________ Office ____________
Memberships - $16.00 (family/individual);make checks payable to C.A.T.S.
If family membership, please list family members participating:
____
____
____
Occupation ____________________
Ham Radio call sign ____________________

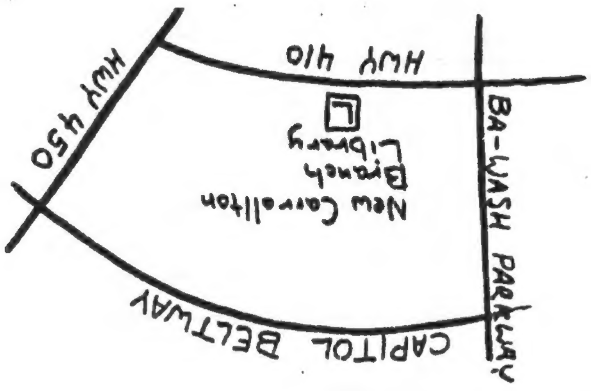
Equipment

ZX 80 ________ RAM size ________
MA 80 ________ full keyboard ________
ZX 81 ________ Printer ________
TS 1000 ________ type ________
TS 2000 ________ other interface ________

Special interest use for computer: ie, games, ham radio interface, business, other, etc. ____________________
____
____
Languages: Basic ________ Other ________
Machine ________
No. of years computer experience ________
What committees would you like to serve on? ____________________
Comments: Where did you hear of C.A.T.S.?
____
____

- - - - - - - - - - - - - - - - - - - -
Do not write below:
Dt. Pd. ________ Amt. ________ Membership No. ________
Ca. ________ Ck. ________ 19

CATS Newsletter
P.O. Box 725
Bladensburg MD 20710



COME TO OUR MEETING!
The next meeting of C.A.T.S. will be held on:
Saturday, August 9, at 2:00 PM

At: New Carrollton Public Library
7414 Riverdale Road (Hwy 410), New Carrollton, MD

IF YOU ARE NOT A MEMBER OF CATS, THIS IS THE ONLY ISSUE YOU WILL RECIEVE

Dues = $16.00 per year, per family.

First Class

The mailing address of the Capitol Area Timex/Sinclair User's Group is:

Capitol Area Timex/Sinclair User's group
P.O. Box 725
Bladensburg, MD 20710

CATS is a non-profit special interest organization dedicated to serving the interests of those who own, use, or are interested in learning more about the Timex/Sinclair family of personal computers.

The official contact person for CATS is JULES GESANG:

301*922-0767

Meetings are held on the second Saturday of each month at 2 P.M. in the large meeting room of the New Carrollton Branch Public Library.

Ham Radio Network Information
QZX Net...Wednesdays, 9p.m. local time; 14.345 MHz NV4F NCS
Eastern Regional Sinclair Net...Sundays, 1600 Z; 7.245 MHz
KQ2F NCS